资源约束项目的改进差分进化参数控制及双向调度算法

项 前[1] 周亚云[1] 吕志军[1]

摘 要 针对资源约束项目调度组合优化难题, 提出一种改进的动态差分进化参数控制及双向调度算法. 通过参数时变衰减与个体优劣评价, 自适应控制个体进化参数, 提高算法的收敛性能、勘探与开发最优解的能力; 基于动态差分进化 (Dynamic differential evolution, DDE), 提出一种双向调度算法, 使用满足任务时序约束的优先数编码、交替正向反向调度, 结合标准化编码调整与精英保留的种群随机重建策略, 建立了一种高效稳健的双向编码调整机制. 通过著名的项目调度问题库 (Project scheduling problem library, PSPLIB) 中实例集测试, 并与其他文献算法比较最优解平均偏差率, 验证了所提算法的有效性与优越性.

关键词 资源约束项目, 动态差分进化, 参数控制, 双向调度

引用格式 项前, 周亚云, 吕志军. 资源约束项目的改进差分进化参数控制及双向调度算法. 自动化学报, 2020, **46**(2): 283−293

DOI 10.16383/j.aas.c170728

Improved Differential Evolution Parameter Control and Bidirectional Scheduling Algorithm for the Resource-constrained Project

XIANG Qian[1] ZHOU Ya-Yun[1] LV Zhi-Jun[1]

Abstract Aiming at the resource-constrained project scheduling problem (RCPSP) which is one of the most intractable combinatorial optimization problems, an improved dynamic differential evolution parameter control method and a bidirectional scheduling algorithm are proposed. Through parameters time-varying attenuation and individual objective evaluation, evolution parameters are adaptively controlled to improve the convergence performance of the algorithm and the ability of exploring and exploiting the optimal solution. Based on the dynamic differential evolution, a bidirectional scheduling algorithm, which uses time precedence constrained activity coding, alternating forward-backward scheduling and combinatorial strategies of standardized coding adjustment and elitist reserved population random reconstruction, is designed to establish an efficient and robust bidirectional coding adjustment mechanism. Through testing the instance set of the wellknown project scheduling problem library (PSPLIB) and comparing the average deviation rate with those of other algorithms, the proposed algorithm shows its effectiveness and superior performance.

Key words Resource-constrained project, dynamic differential evolution, parameter control, bidirectional scheduling

Citation Xiang Qian, Zhou Ya-Yun, Lv Zhi-Jun. Improved differential evolution parameter control and bidirectional scheduling algorithm for the resource-constrained project. *Acta Automatica Sinica*, 2020, **46**(2): 283−293

资源约束项目调度问题 (Resource-constrained project scheduling problem, RCPSP) 是指在满足项目资源约束及项目任务开始时间顺序约束的前提下, 以项目的最小工期为目标, 安排最优的任务开始时间. RCPSP 广泛存在于企业的设计、制造、生产管理等活动中, 属于组合优化难题, 倍受国内外学者关注.

收稿日期 2017-12-25 录用日期 2018-07-05
Manuscript received December 25, 2017; accepted July 5, 2018
国家重点研发计划 (2017YFB1304000), 上海市科学技术委员会科研计划项目 (17DZ2283800) 资助
Supported by National Key R & D Program of China (2017YFB1304000), Research Program of Shanghai Science and Technology Commission (17DZ2283800)
本文责任编委 付俊
Recommended by Associate Editor FU Jun
1. 东华大学机械工程学院 上海 201620
1. College of Mechanical Engineering, Donghua University, Shanghai 201620

目前求解 RCPSP 的算法主要分三类[1]: 适用于小规模、复杂度低的精确算法; 基于调度优先规则的启发式算法, 效率高但质量难以保证; 具有较好鲁棒性的元启发式算法, 如遗传算法、粒子群算法、蚁群算法、模拟退火、禁忌搜索、差分进化等, 适合于求解大规模组合优化问题. 多年来, 以进化与群体智能为主的 RCPSP 元启发式算法研究不断涌现, 在个体编码表示、调度方案生成、进化机制改进、与其他算法结合使用等方面, 国内外学者进行了大量的研究, 并且积累了丰富的问题实例, 形成了著名的项目调度问题标准库 (Project scheduling problem library, PSPLIB)[2]. Hartmann[3] 较早地基于遗传算法, 研究了任务染色体表示、自适应进化参数控制及调度方法. 邓林义等[4] 通过优先规则粒子群表示与更新机制, 搜索优先规则和调度方案的最优组

合. 戴月明等[5] 使用粒群算法, 通过优先数粒子表示与双向协同震荡搜索机制, 提升调度算法性能. Gonzalez-Pardo 等[6] 基于蚁群算法, 结合珊瑚礁优化算法控制搜索过程中产生的信息素, 取得满意的调度解. Munlin 等[7] 通过适应性变异与双向调整[8], 提出一种求解 RCPSP 的扩展粒子群算法 (Hybrid radius particle swarm optimization, H-RPSO). Zamani[9] 结合隐式枚举搜索加速技术, 使用遗传算法求解 RCPSP. Paraskevopoulos 等[10] 通过存储每个任务的调度历史, 提出一种具有学习机制的响应式变量邻近搜索算法 (Reactive variable neighbourhood search, ReVNS). 黄志宇[11] 将调度解量子表示方法与进化算法融合, 有效地提高了全局寻优能力. 陆志强等[12] 建立考虑资源转移时间 RCPSP 模型, 提出内嵌分支定界法的遗传算法, 在保证全局搜索能力的前提下提升局部搜索能力. Koulinas 等[13] 提出的 RCPSP 超启发式算法与 Elsayed 等[14] 提出的组合式优化算法, 灵活地发挥了多种元启发式算法的优势, 算法在高层自适应优选元启发式算法种类及其算子, 在低层使用具体的元启发算法进行求解, 提高了算法的鲁棒性与解的质量.

除了以上算法以外, 近年来, 差分进化算法 (Differential evolution algorithm, DE) 因其卓越的优化性能受到广泛关注, 其应用范围越来越广, 研究问题主要集中在单目标约束与无约束、大规模、多目标、动态优化以及与其他优化算法混合使用等[15−16]. 研究显示, DE 算法已逐渐成为 RCPSP 研究的有力工具之一, Cheng 等[17] 在 DE 中使用混沌技术避免种群早熟, 使用模糊聚类技术加速算法收敛, 通过 2 个小规模任务案例测试, 验证了算法的有效性. Ali 等[18] 提出了一种个体优先数编码调整及改进变异操作的 DE 算法, 通过项目调度标准库 PSPLIB 中部分实例测试, 验证了算法的优越性能. 然而, 与遗传算法、粒子群算法相比, 运用 DE 进行 RCPSP 的研究还相对较少[19]. 目前 DE 的相关理论分析还不够完善, 缺乏严谨的数学解释, 算法的性能优劣很大程度上取决于其控制参数的选择, 标准 DE 中, 通常依赖经验选择固定的变异缩放因子 F 与交叉概率 CR 等, 难以应对如 PSPLIB 中的不同实例问题. 长期以来, DE 进化参数适应性控制与算子改进一直是研究热点[16], Brest 等[20] 提出改进 DE 算法, 引入两个控制参数 τ_1, τ_2 来随机调整 F 和 CR 的值. Qin 等[21] 提出 SaDE (Self-adaptive differential evolution) 算法, 参考历史优质解的经验, 自适应选择变异策略, 自适应调整服从正态分布的 F 和 CR, 平衡了算法对优质解的搜索与开发能力. Zhang 等[22] 提出 DE/current-to-pbest 变异策略, 建立了较差解的外部存档, 进化中自适应更新 F 与 CR.

综上所述各种元启发式算法, 在算法机制改进方面, 通常需要对算法参数或操作算子进行适应性调整, 或者与其他算法组合使用; 在个体编码方面, 需要充分利用问题自身的特征, 以减少对编码空间的贪婪搜索, 提高算法的收敛性能与求解质量. 因此, 针对 RCPSP, 本文提出一种改进的动态差分进化参数控制及双向调度算法, 通过参数时变衰减与个体优劣评价, 自适应控制个体进化参数, 设计满足时序约束的任务优先数编码与双向调度算法, 建立避免算法早熟的种群重建机制, 以提高算法的求解质量与收敛性能. 通过 PSPLIB 中实例集测试与分析, 验证所提算法有效性.

1 问题模型

经典 RCPSP 中, 单个项目包括 n 个任务或活动, 记 s_i 为任务 i 的开始时间, d_i 为任务 i 的消耗工时 $(i = 1, 2, \cdots, n)$, r_{ik} 为任务 i 对第 k 类资源的需求量, R_k 为第 k 类资源的总量 $(k = 1, 2, \cdots, K)$, 任务 1 和任务 n 是虚拟任务, 分别代表项目的开始和结束, 且不消耗时间和资源, 即 $s_1 = 0$, $d_1 = d_n = 0$, $r_{1k} = r_{nk} = 0$. 对应的数学优化模型为

$$\boldsymbol{S} = (s_1, s_2, \cdots, s_n) \tag{1}$$

$$\min\ f(\boldsymbol{S}) = \max\{e_i\},\ e_i = s_i + d_i,\ i = 1, 2, \cdots, n \tag{2}$$

$$\text{s.t. } s_i \geq s_j + d_j,\ j \in P_i,\ i = 1, 2, \cdots, n \tag{3}$$

$$\sum_{i \in A(t)} r_{ik} \leq R_k,\ k = 1, 2, \cdots, K,\ t \geq 0 \tag{4}$$

其中, 式 (1) $\boldsymbol{S}$ 为各任务开始时间 s_i 组成的解向量; 式 (2) f 为目标函数, 表示使所有任务完成的总工期最小, e_i 为任务 i 的结束时间; 式 (3) 表示各任务开始时间顺序约束, 表示任务 i 必须在其所有前序任务 j 完成后才能开始执行, P_i 为任务 i 的前序任务集合, s_j 为任务 j 的开始时间, d_j 为任务 j 的消耗工时; 式 (4) 为资源约束, 表示任务 i 在执行中任一时刻 t 对资源 k 的使用量不超过许可用量 R_k, 其中 $A(t)$ 为 t 时刻进行的任务集合.

2 动态差分进化参数自适应控制

动态差分进化 (Dynamic differential evolution, DDE) 参数自适应控制方法包括两点: 基于动态差分进化机制, 每代进化中及时更新目标个体动态产生新种群, 可加速算法收敛; 综合考虑参数衰减与个

体目标值优劣评价, 自适应控制个体变异与交叉参数, 能有效控制种群的搜索与开发能力.

2.1　动态差分进化

标准差分进化算法中, 目标个体 i 表示为矢量 $\boldsymbol{X}_i^t = (x_{i1}^t, x_{i2}^t, \cdots, x_{in}^t)$, $i = 1, 2, \cdots, NP$, 其中, NP 为种群规模, n 为优化变量个数, t 为进化代数. 随机初始化种群后开始迭代进化过程, 如式 (5) 所示, 第 t 代目标个体 $\boldsymbol{X}_i^t$ 进行变异生成变异矢量 $\boldsymbol{V}_i^t = (v_{i1}^t, v_{i2}^t, \cdots, v_{in}^t)$, 其中, 下标索引 r_1, r_2, r_3 为从集合 $\{1, 2, \cdots, NP\}$ 中随机选择的互不相同的整数, 且不等于目标个体的索引 i, 缩放因子 F 作用是对扰动量 $\boldsymbol{X}_{r_2}^t - \boldsymbol{X}_{r_3}^t$ 进行比例缩放, 即控制个体的搜索范围.

$$\boldsymbol{V}_i^t = \boldsymbol{X}_{r_1}^t + F(\boldsymbol{X}_{r_2}^t - \boldsymbol{X}_{r_3}^t) \tag{5}$$

如式 (6) 所示, 变异矢量 $\boldsymbol{V}_i^t$ 与个体 $\boldsymbol{X}_i^t$ 使用二项式交叉产生试验矢量 $\boldsymbol{U}_i^t = (u_{i1}^t, u_{i2}^t, \cdots, u_{in}^t)$, 使得试验矢量至少有一个维度分量由变异矢量 $\boldsymbol{V}_i^t$ 贡献, 其中, $j = 1, 2, \cdots, n$, 随机整数 $j_{\text{rand}} \in [1, n]$, 交叉率 $CR \in (0, 1)$.

$$u_{ij}^t = \begin{cases} v_{ij}^t, & \text{rand}(0,1) \le CR \text{ 或 } j = j_{\text{rand}} \\ x_{ij}^t, & \text{否则} \end{cases} \tag{6}$$

如式 (7) 所示, 个体选择阶段, 比较试验矢量 $\boldsymbol{U}_i^t$ 与目标个体矢量 $\boldsymbol{X}_i^t$ 的目标函数值, 挑选最优的进入下一代.

$$\boldsymbol{X}_i^{t+1} = \begin{cases} \boldsymbol{U}_i^t, & f(\boldsymbol{U}_i^t) \le f(\boldsymbol{X}_i^t) \\ \boldsymbol{X}_i^t, & \text{否则} \end{cases} \tag{7}$$

标准 DE 每一代使用额外的存储空间缓存选择的个体, 当完成种群内所有个体的选择操作后, 才集体更新当前种群进入下一代. 与标准 DE 的种群更新方式不同, 动态差分进化策略[23] 选择的新个体及时替代旧目标个体, 并加入当前种群参与其他个体的进化. 这种动态更新方式能及时利用新的进化成果, 从而加速了算法收敛.

2.2　考虑时变特性与个体优劣的参数控制

如式 (8) 所示, 给出了一种自适应参数控制方法, 在进化过程中考虑了参数时变衰减特性, 在种群内基于个体目标值的优劣, 对个体参数进行差异化调整与控制.

$$P_i(t) = P_l + (P_u - P_l)[\lambda \cdot \alpha(t) + (1-\lambda)\beta_i], \quad 0 < \lambda < 1 \tag{8}$$

$$\alpha(t) = \frac{T - t}{T} \tag{9}$$

$$\beta_i = \begin{cases} \frac{f_i - f_{\min}}{f_{\max} - f_{\min}}, & f_{\max} \ne f_{\min} \\ 1, & \text{否则} \end{cases} \tag{10}$$

式 (8) ∼ (10) 中, P_i 表示个体 i 参数 F 或 CR; P_u 与 P_l 分别代表参数 F 或 CR 值的上限与下限; $\alpha(t)$ 是个体参数 P_i 按进化代数 t 线性衰减的系数; T 为允许最大进化代数; β_i 表示按目标函数值评价个体 i 的优劣程度, $\beta_i \in [0, 1]$, β_i 越小表示个体越优, 反之越差; λ 是对 $\alpha(t)$ 与 β_i 加权平均的权重因子, 本文取 $\lambda = 0.5$; f_i 为个体 i 目标函数值, $f_{\min}$ 为种群内最优 (小) 个体目标函数值, $f_{\max}$ 为种群内最差 (大) 个体目标函数值.

从进化过程来看, 种群进化初期一般希望具有较好的全局搜索能力, 而随着迭代次数的增加, 特别是在后期, 希望具有较好的局部开发能力. 较大的 F 产生扰动量大, 会扩大搜索范围, 较大的 CR 会增加种群的多样性, 有利于全局搜索, 反之, 较小的 F 会缩小搜索范围, 较小的 CR 会减少种群的多样性, 这样虽会减缓进化, 但有利于稳定的局部开发. 因此, 通过式 (9) 中的 $\alpha(t)$, 使用时变衰减的 F 与 CR, 可以平衡搜索与开发能力.

从个体在种群内的优劣表现来看, 在以最小化为目标的前提下, 目标函数值越小表示个体越优秀, 当个体目标函数值较小时, 希望算法更多地利用该优秀个体的信息进行局部开发, 需要使用较小的 F, 在该个体附近搜索较好解, 且使用较小的 CR, 增加当前优质解进入下一代的机会; 当个体目标函数值较大时, 应较少利用该个体信息, 使用较大的 F 以扩大搜索范围, 增加获得更优解的机会, 且使用较大的 CR, 促进新个体结构产生, 增加种群多样性. 因此, 通过式 (10) 中的 β_i, 个体根据自身目标函数值的优劣自适应调整参数, 有利于进化过程稳健地进行.

根据以上分析, 与经典的参数控制方法相比, 考虑时变特性与个体优劣的自适应参数控制方法, 使种群获得稳健的搜索与开发能力, 减少了选择参数时的经验依赖性以及搜索时的盲目性, 有利于改进算法收敛性能.

3　基于 DDE 的双向调度算法

基于动态差分进化 (DDE), 提出一种双向调度算法 (Bidirectional scheduling algorithm), 建立了高效鲁棒的双向编码调整机制. 使用满足时序约束的个体编码表示, 生成初始种群; 在交替的正向反向调度中引入标准化编码调整策略, 以减少冗余搜索并加速算法收敛; 采用精英保留的种群随机重建策略避免算法陷入局部最优, 进一步提高算法的求解质量与鲁棒性.

3.1 满足时序约束的个体编码

初始种群采用随机数编码, 目标个体 $\boldsymbol{X}_i$ 中任务 k 的初始码值 x_{ik}, $k = 1, 2, \cdots, n$ 为 0 到 1 之间的随机数, 其大小表示任务 k 的调度优先级, 越小表示越优先, 通过码值升序排列生成任务调度顺序. 为了满足任务开始时间顺序约束, 需要对初始编码进行调整, 使任务 k 的调度优先级高于其所有后继任务. 以图 1 为例, 任务有向图 G 包括 6 个任务结点, 初始任务 1 和结束任务 6 为虚拟任务, 不消耗时间和资源, 对应于式 (3) 时序约束条件, 有向边表示任务之间的时序依赖关系, 根据 RCPSP 问题描述, 有向图 G 中不存在任务循环依赖, 例如, 任务 2 的直接或间接后续任务是 4、5、6.

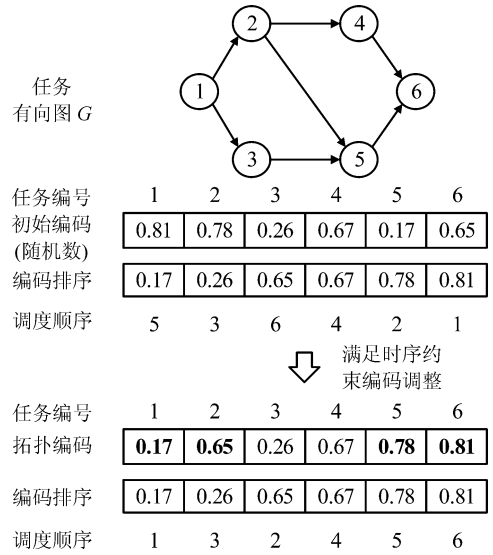


图 1 个体编码与任务调度顺序

Fig. 1 Individual coding and task scheduling sequence

算法 1. 满足时序约束的编码调整算法

输入: n 个任务的有向图 G; 个体 i 编码 $\boldsymbol{X}_i = \{x_{ik} | k = 1, 2, \cdots, n\}$

输出: 拓扑编码个体向量 $\boldsymbol{X}_i$

1: For 任务 $k = 1$ to n do
2: 初始化任务 k 的所有后继任务集合 $S = \emptyset$ (空集), 集合 $T = \{k\}$
3: Repeat
4: 初始化临时空集合 $P = \emptyset$
5: For 每个任务 $m \in T$ do
6: 从 G 中搜索直接依赖于任务 m 的任务集合 S_m
7: $P \leftarrow P \cup S_m$
8: End For
9: $S \leftarrow S \cup P, T \leftarrow P$
10: Until $T = \emptyset$
11: 查找具有最小编码值的任务 $j \in S$, $x_{ij} = \min\{x_{ir} | r \in S\}$
12: If $x_{ik} > x_{ij}$ then 交换 x_{ik} 与 x_{ij}
13: End For
14: Return $\boldsymbol{X}_i$

个体用随机数数组表示, 任务编号对应于数组索引号. 初始化种群时, 如果直接按照编码升序来重排任务编号, 那么生成的任务调度序列 5-3-6-4-2-1 将违背时序约束. 因此, 在个体初始随机编码的基础上, 根据时序约束使用算法 1 进行编码调整, 形成满足时序约束的个体拓扑编码及任务调度序列 1-3-2-4-5-6. 除此以外, 变异与交叉操作产生的试验矢量, 由于随机性也可能不满足时序约束, 同样需要编码调整.

3.2 标准化编码调整

从编码的随机性可以看出, 不同个体编码可能对应于同一个调度顺序, 即产生同一个调度解, 进化过程中会出现冗余搜索, 减缓寻优进程. 因此, 在种群进化过程中, 采用一种简易的标准化编码调整策略, 具体思路是, 在个体评价产生调度可行解后, 对可行解中任务开始时间排序, 获得相应的任务执行顺序, 记任务 i 的顺序号为 k ($k = 1, 2, \cdots, n$), 则任务 i 的标准化编码值为 k/n, 再使用此标准化编码调整原个体编码. 该算法的优点是, 使得一个调度顺序或调度解对应于一种个体编码模式, 从而在进化过程中会不断缩小种群搜索空间, 加速 DE 算法的收敛. 存在的缺点是, 个体标准化编码会降低种群的多样性, 易导致算法收敛早熟, 这一缺点会通过第 3.4 节中的技术得以补偿, 第 4.1 节中将对其效果进行验证.

3.3 双向编码调整

调度算法实质是对个体评价的过程, 即将编码转化为任务开始时间与项目总工期. 首先, 个体编码映射为一个任务调度序列, 其次, 在满足式 (3) 与式 (4) 约束条件下, 使用串行调度策略逐个安排任务的开始时间, 最终, 按式 (2) 目标函数求得项目总工期.

正向调度 (Forward scheduling) 指先安排前序任务再安排后继任务的调度方式. 如图 1 所示, 从虚任务 1 开始, 设置开始时间为 $s_1 = e_1 = 0$, 根据任务调度顺序列 1-3-2-4-5-6, 依次追尾插入每个任务 i ($i = 3, 2, 4, 5, 6$), 在满足时序与资源约束的前提下, 安排任务 i 的最早开始时间 $s_i = s_k + d_k$, 即任务 i 的某个紧前任务 k 的结束时间, 插入完毕后, 计算项目总工期为 $\max\{e_i = s_i + d_i\} - s_1$, 开始时间 s_i 按照任务编号升序排列成的向量, 生成对应于个体 m 的一个可行调度解 $\boldsymbol{S}_m = (s_{m1}, s_{m2}, \cdots, s_{mn})$.

反向调度 (Backward scheduling) 与正向调度

的求解方向相反, 仍以图 1 问题为例, 反向调度从调度序列 1-3-2-4-5-6 中最后一个任务 $n = 6$ 开始, 设置任务 n 的结束时间 $e_n = s_n = 0$, 在满足时序与资源约束的前提下, 倒序逐个安排任务 i ($i = 5, 4, 2, 3, 1$) 的最晚结束时间 $e_i = s_k$, 即任务 i 的某个紧后任务 k 的开始时间, 从而求得任务 i 开始时间 $s_i = e_i - d_i$, 考虑到 s_i 会出现负值, 在获得所有任务开始时间后, 通过 $s_i = s_i - \min\{s_i\}, i = 1, 2, \cdots, n$, 对 s_i 进行修正使得 $s_i \geq 0$. 除调度顺序引起的变化之外, 反向调度中目标总工期及可行调度解的算法与正向调度相同.

根据 Valls 等[8] 研究, 交替正向与反向调度, 通过双向编码调整可开发出更优的解, 但这样也意味着每个个体评价耗时更多. 因此, 在交替正向反向调度中引入标准化编码调整策略, 从而兼顾求解质量与效率. 基本思路如算法 2 所述, 通过交替进行正向与反向调度算法, 每次正 (反) 向调度完成后, 使用标准化编码方法, 将调度生成的开始时间解转化为标准化编码, 临时存储该标准码, 作为下一轮反 (正) 向调度的个体编码输入, 重复以上过程, 如果交替求解中能得到改进解, 即更短的项目总工期, 那么继续正反向交替调度, 否则退出循环, 在循环结束时, 使用标准化编码更新原个体码值.

算法 2.　双向编码调整算法

输入: 个体编码矢量 $\boldsymbol{X}_i^t$

输出: 调度解 $\boldsymbol{S}_i^t = (s_1^t, s_2^t, \cdots, s_n^t)$ 以及总工期 (目标值 *objective*)

1: 初始化计数变量 $k = 0$, 临时向量 $\boldsymbol{p} = \boldsymbol{X}_i^t$, 目标值 $objective = +\infty$
2:　Repeat　do
3:　　If　k 为偶数　then
4:　　forward($\boldsymbol{p}, out\ \boldsymbol{S}_i^t, out\ objVal$)//正向调度
5:　　Else
6:　　backward($\boldsymbol{p}, out\ \boldsymbol{S}_i^t, out\ objVal$)//反向调度
7:　　End　If
8:　　If　$objVal \geq objective$　then　break
9:　　$objective = objVal$
10:　$\boldsymbol{p}$ = standardize($\boldsymbol{S}_i^t$) //参见第 3.2 节标准化编码
11:　$k++$
12: End　Repeat
13: $\boldsymbol{X}_i^t = \boldsymbol{p}$ // 个体编码标准化更新

3.4　避免陷入局部最优

动态差分进化到一定代数后, 个体的多样性会下降易产生早熟现象. 为了避免算法过早地停滞于局部最优解, 需要增加种群的多样性扩大搜索空间, 提高全局寻优能力. 避免陷入局部最优的策略是, 在不超过最大迭代代数的范围内, 当种群个体平均目标函数值 f_{avg} 与种群最小目标函数值 $f_{\min}$ 之间差异 $\varepsilon = f_{\text{avg}} - f_{\min}$ 很小, 而且最优目标值持续未改进超过一定代数时, 保留部分当前精英个体, 如保留排名前 10 % 的优质个体, 对其他个体进行随机编码重建, 使得大部分个体逃离原先的生存地开辟新的搜索进程, 从而提高了算法的全局寻优能力与鲁棒性.

3.5　算法流程

本文算法的基本流程如图 2 所示, 具体步骤如下:

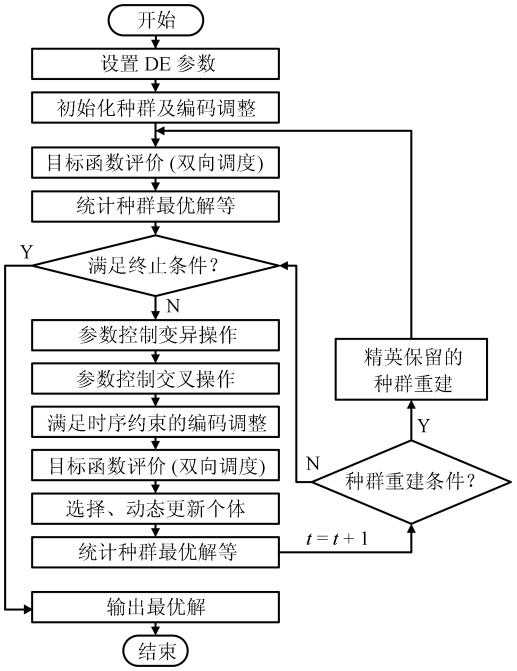


图 2　动态差分进化参数控制及双向调度算法流程图

Fig. 2　Flowchart of DDE parameter control and bidirectional scheduling algorithm

步骤 1. 设置种群规模为 NP, 允许最大进化代数为 T, 初始迭代计数器变量 $t = 0$; 初始化种群, 个体矢量的维数等于调度任务数 n, 每个个体先进行随机数编码, 再采用算法 1 进行编码调整;

步骤 2. 按照第 3.3 节中算法 2, 计算种群中每个个体 $\boldsymbol{X}_i^t$ 的目标函数值 $f(\boldsymbol{X}_i^t)$, 并获得对应的调度任务开始时间解 $\boldsymbol{S}_i^t$, 根据目标值大小, 统计种群获得当前最优个体 $\boldsymbol{X}_{\text{best}}^t$、调度解 $\boldsymbol{S}_{\text{best}}^t$ 及最优目标值 $f_{\min}$ (即项目最短总工期, 以下亦简称为最优解);

步骤 3. 开始循环迭代, 判断是否满足终止条件, 即如果 t 大于 T, 或者达到已知最优解 (适用于已知最优解的测试实例), 或者全局最优解停滞未改

进代数超过了上限 M_1 (适用于最优解未知的问题, 如 $M_1 = 300$ 代), 则退出循环至步骤 8, 否则, 进行下一步;

步骤 4. 对种群中每一个个体 $\boldsymbol{X}_i^t$ 循环执行步骤 4.1∼4.4, 生成第 $t+1$ 代种群;

步骤 4.1. 从种群中随机抽取 3 个不同个体, 按式 (8) 获得动态自适应缩放因子 F_i, 按式 (5) 进行变异操作, 生成变异个体 $\boldsymbol{V}_i^t$;

步骤 4.2. 按式 (8) 获得动态自适应交叉概率 CR_i, 按式 (6) 进行交叉操作, 生成试验个体 $\boldsymbol{U}_i^t$;

步骤 4.3. 采用算法 1 对试验个体 $\boldsymbol{U}_i^t$ 进行编码调整, 并且计算目标函数值 $f(\boldsymbol{U}_i^t)$, 获得调度解 $\boldsymbol{S}_i^t$;

步骤 4.4. 按照式 (7) 进行选择操作生成下一代个体 $\boldsymbol{X}_i^{t+1}$, 动态更新个体 $\boldsymbol{X}_i^t$;

步骤 5. 根据目标值大小, 统计种群获得最优个体 $\boldsymbol{X}_{\text{best}}^{t+1}$、调度解 $\boldsymbol{S}_{\text{best}}^{t+1}$、种群最优目标值 $f_{\min}$、以及种群平均目标函数值 f_{avg};

步骤 6. 如果 $f_{\text{avg}} - f_{\min} \le \varepsilon$, $\varepsilon = 0.1$, 且停滞未改进代数达到上限 M_2 ($M_2 < M_1$, 如 $M_2 = 100$ 代), 则按第 3.4 节算法保留前 10 % 精英个体, 其余个体重新随机创建, 并且重新评价和统计种群, 否则, 维持种群不变.

步骤 7. 令 $t = t + 1$, 转步骤 3.

步骤 8. 输出当前种群最优个体 $\boldsymbol{X}_{\text{best}}^t$、调度解 $\boldsymbol{S}_{\text{best}}^t$ 及最优目标值 $f_{\min}$.

4 算法试验及分析

为了验证本文所提自适应 DDE 参数控制与双向调度等算法, 采用项目调度标准库 PSPLIB 中 2 040 个 RCPSP 实例, 测试算法效果. PSPLIB 按照项目任务数 30、60、90、120, 分别设置了 J30 (480 个实例)、J60 (480 个实例)、J90 (480 个实例)、J120 (600 个实例) 四个系列. 鉴于仅 J30 使用精确算法给出了全部最优解, 而 J60、J90、J120 只给出了部分实例的最优解下界, 因此, 先使用 J30 实例集测试所提算法有效性, 再使用 J30、J60、J90、J120 验证与比较算法性能.

4.1 试验方案比较分析

如表 1 所示, 设计了 8 种测试方案, 统一设置种群规模为 50, 允许最大进化代数 $T = 1\,000$, 即最大调度求解次数不超过 50 000; 最大停滞代数 $M_1 = 1\,000$, $M_2 = 100$; 除方案 1 使用标准 DE 以外, 其余方案 2 ∼ 8 都使用动态差分进化 (DDE); 方案 1 ∼ 2 采用相同的固定参数, 设置缩放因子 F 为 0.5、交叉概率 CR 为 0.5; 方案 3 参照文献 [21] 的参数控制算法, 缩放因子 F 服从均值为 0.5、标准差为 0.3 的正态分布 N $(0.5, 0.3^2)$, 交叉概率 CR 服从均值为 0.5、标准差为 0.1 的正态分布 N $(0.5, 0.1^2)$; 方案 4 ∼ 8 采用本文提出的参数自适应调整方法, 缩放因子 $F = 0.1 \sim 2.0$, 交叉概率 $CR = 0.1 \sim 0.95$. 试验计算机配置为 Intel®Core™/i5CPU/2.2 GHZ/4 GB 内存, Windows10 操作系统, 使用 C#、Matlab 作为编程及分析工具.

类似于其他 RCPSP 研究中算法性能测评方法, 对于单个实例测试, 主要使用偏差率 dev_i 评价算法准确性; 对于标准实例集系列, 主要使用平均偏差率 av_dev、成功率 $success$ 评价算法性能. 每个试验方案中, 每实例独立测试次数 $J = 10$.

$$dev_i = \frac{BS_i - LB_i}{LB_i} \times 100\,\% \tag{11}$$

$$av\_dev = \frac{1}{N}\sum_{i=1}^{N} dev_i \times 100\,\% \tag{12}$$

$$success = \frac{1}{J}\sum_{j=1}^{J} \frac{C_j}{N} \times 100\,\% \tag{13}$$

式 (11) ∼ (13) 中, 偏差率 dev_i 是单个实例 i 目标解与已知最优解之间的相对误差, 平均偏差率 av_dev 是测试实例集中所有偏差率的平均值, 成功率 $success$ 是求解正确的实例数百分比. 其中, $i = 1, 2, \cdots, N$, N 为测试实例总数, 如测试 J30 时 $N = 480$; BS_i 为求解实例 i 的目标最短工期; 值得注意的是, 对于 J30 系列, LB_i 为实例 i 已知最优解, 对于 J60、J90、J120 系列, LB_i 为已给出的实例 i 最优解或下界, 若尚未给出最优解或下界, LB_i 则取无资源约束时实例 i 的最短工期; C_j ($j = 1, 2, \cdots, J$) 为第 j 次测试时 BS_i 等于 LB_i 的实例数. 除以上性能指标以外, 使用平均代数 av_gen 与平均 CPU 时间 $CPUTime$, 即平均每实例每次测试所需进化代数与运行时间, 来评价算法效率.

如表 1 所示, 平均偏差率、成功率、平均代数及平均 CPU 时间等测试指标值, 反映了不同试验方案的算法性能. 为了检验任意两个不同方案间性能差异是否显著, 针对关键性能指标平均偏差率, 设计双总体 t 检验, 显著性水平 $\alpha = 0.05$, 零假设 H0 为两者平均偏差率相同, 备择假设 H1 为两者平均偏差率不同. 如表 2 所示, 三角矩阵行列序号表示方案代号, 单元格值为接受零假设 H0 的概率, 概率值越大表示对应两个方案间差异越小, 概率值越小表示对应两个方案间具有显著差异.

分析各方案试验结果, 从表 1 ∼ 2 与图 3 中可以看出:

表 1　J30 系列不同测试方案结果比较

Table 1　Comparison of the results by different test schemes for J30

方案	DE 类型	参数控制	调度算法	避免局部最优	标准化编码	平均偏差率 av_dev (%)		成功率 $success$ (%)	平均代数 av_gen	平均 CPU 时间 $CPUTime$ (s)
						平均值	标准差			
1	DE	固定	正向			0.143	0.013	92.40	109	4.59
2	DDE	固定	正向			0.143	0.021	92.42	90	2.54
3	DDE	正态分布[21]	正向			0.142	0.015	92.46	89	2.62
4	DDE	自适应	正向			0.050	0.011	96.94	53	1.56
5	DDE	自适应	双向			0.027	0.005	98.25	27	2.22
6	DDE	自适应	双向		是	0.065	0.013	96.00	44	1.14
7	DDE	自适应	双向	是		0.016	0.005	98.90	22	1.91
8	DDE	自适应	双向	是	是	0.008	0.003	99.35	13	0.93

表 2　显著性水平 ($\alpha = 0.05$) 下平均偏差率双总体 t 检验

Table 2　Paired-sample t-test of the average deviation rate at the significance level of 0.05

	1	2	3	4	5	6	7	8
1	1							
2	**0.983**	1						
3	**0.939**	**0.937**	1					
4	1.34×10^{-12}	3.83×10^{-10}	4.66×10^{-12}	1				
5	6.69×10^{-16}	1.98×10^{-12}	4.33×10^{-15}	1.38×10^{-5}	1			
6	8.97×10^{-11}	1.16×10^{-8}	2.63×10^{-10}	1.18×10^{-2}	8.02×10^{-8}	1		
7	1.30×10^{-16}	4.06×10^{-13}	8.40×10^{-16}	5.81×10^{-8}	3.64×10^{-5}	1.50×10^{-9}	1	
8	$\mathbf{2.83\times10^{-17}}$	$\mathbf{1.25\times10^{-13}}$	$\mathbf{2.03\times10^{-16}}$	$\mathbf{1.53\times10^{-9}}$	$\mathbf{4.69\times10^{-9}}$	$\mathbf{9.62\times10^{-11}}$	$\mathbf{5.62\times10^{-4}}$	1

1) 比较方案 1、2、3, 三者的平均偏差率、成功率基本相同或相近, 表 2 中三者之间 t 检验接受零假设的概率值都超过 92 %, 从运行效率来看, 方案 2、3 的平均代数与平均 CPU 时间接近, 比方案 1 的效率高. 由此表明, DDE 与 DE 相比, 随机调整参数与固定参数相比, 都不能降低平均偏差率与提高成功率, 但方案 2、3 中 DDE 有助于加速算法收敛.

2) 方案 4 与方案 1 ∼ 3 相比, 同样使用正向调度, 方案 4 使用了本文的 DDE 自适应参数控制方法. 从求解质量来看, 方案 4 平均偏差率为 0.05 %, 成功率达到 96 % 以上, 显著优于方案 1 ∼ 3; 从获得最优解平均代数以及平均 CPU 时间来看, 方案 4 也比方案 1 ∼ 3 效率高. 相比于方案 1 ∼ 3 的固定或随机调整参数策略, 方案 4 采用控制参数时变衰减, 平衡了全局搜索与局部开发能力; 基于个体目标值优劣评价, 实现个体参数差异化设置, 使个体能够更合理有效地搜索, 一定程度地克服了参数设置的经验性与个体搜索时的盲目性, 因此, 算法性能获得了显著提高.

3) 方案 5 与方案 4 相比, 除调度算法不同之外, 其他参数设置相同. 方案 5 使用交替正向反向调度, 平均偏差率为 0.027 % 与成功率为 98.25 %, 均显著优于方案 4. 获得最优解的平均代数比方案 4 减少了约 50 %, 加快了进化的进程, 由于每个个体双向调整比正向调度消耗更多的时间, 算法平均 CPU 时间比方案 4 稍高.

4) 相比于方案 5, 方案 6 在个体调度结束时运用了个体标准化编码调整, 平均偏差率为 0.065 %, 成功率为 96.00 %, 求解质量下降, 但时间却大幅减少. 究其原因, 个体标准化编码调整减少了个体编码结构模式, 收缩了搜索空间, 有利于提高寻优效率, 但明显的弱点是易导致算法局部收敛, 部分测试实例的优化进程出现停滞现象, 因而导致最优解平均代数增加.

5) 相比于方案 5, 方案 7 采用避免局部最优的策略, 通过触发种群重建与搜索重试, 获得平均偏差率 0.016 %, 成功率 98.90 %, 求解效率 (平均 22 代, 1.91 s). 方案 7 性能优于方案 5 的主要原因是, 当种群多样性极低且未达到已知最优解时, 方案 5 会出现搜索停滞直至用尽最大进化代数; 同样条件下, 方

案 7 可能触发种群重建, 对种群进行大幅扰动, 增加了种群多样性与找到最优解的机会, 迭代可能会在达到最大进化代数之前结束, 因此, 方案 7 有利于提高实例集整体求解质量与效率. 方案 7 的平均偏差率比方案 6 低, 但由于没有使用标准化编码调整, 搜索效率比方案 6 略低 (用时 1.91 s).

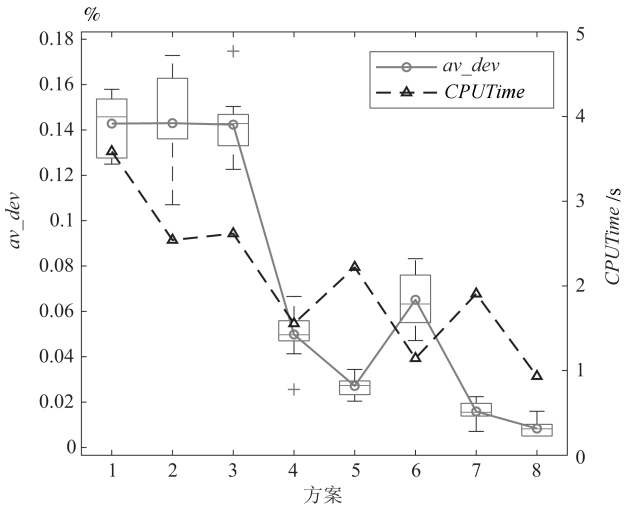


图 3 J30 系列不同 DE 方案性能比较

Fig. 3 Comparison of the performance by different DE schemes for J30

6) 进一步分析避免局部最优策略的效果, 统计试验数据发现, 方案 7 共有 36 个实例 165 次测试触发了种群重建机制, 这 36 个实例相对较难求解或解质量不稳定, 将其归为第 I 类, 第 I 类共 $36 \times 10 = 360$ 次测试, 其中约 $165/360 = 46\,\%$ 的样本触发了种群重建, 其余 444 个实例归为第 II 类. 如表 3 所示, 按照 I、II 类别比较方案 5 与 7 的求解性能, 在第 I 类比较中, 方案 7 获得比方案 5 更低的平均偏差率与平均代数, 表明种群重建机制取得成效; 在第 II 类比较中, 实例相对容易求解, 方案 5 与 7 均无种群重建, 因而两者获得相似的优越性能. 然而, 由于第 I 类测试占总样本的比例过小 ($165/4\,800 = 3.4\,\%$), 因此, 就总体平均代数而言, 表 1 中方案 7 (22 代) 比方案 5 (27 代) 略少, 而且两者的平均代数都较少.

7) 从表 1、3 与图 3 可看出, 方案 8 综合了方案 6 与 7 的优势, 采用方案 6 中标准化编码调整策略, 减少了平均 CPU 时间, 兼用方案 7 中避免局部最优策略, 一定程度地弥补了标准化编码调整的不足, 提高了解的质量. 方案 8 组合了文中所有的算法, 获得了最低的平均偏差率均值 (0.008 %) 及标准差 (0.005 %), 最高的成功率 (99.35 %), 以及最高的求解效率 (平均 13 代, 0.93 s). 由表 2 也可见, 检验方案 8 与方案 1 ∼ 7 平均偏差率差异显著, 均以极低概率接受零假设 H0. 由此表明, 方案 8 算法的准确性与稳定性显著优于其他方案.

表 3 方案 5 与方案 7 性能比较

Table 3 Comparison between scheme 5 and scheme 7

	I 类 (36 个实例)		II 类 (444 个实例)	
	平均偏差率 av_dev (%)	平均代数 av_gen	平均偏差率 av_dev (%)	平均代数 av_gen
方案 5	0.346	285	0.001	6
方案 7	0.194	238	0.001	5

4.2 算法收敛性能分析

为了观察与比较不同算法方案的收敛性能, 选择难度较高的 j3013 组 10 个问题实例, 分别测试方案 1、4、5、8 的算法性能, 每实例独立运行 10 次.

如图 4 所示, 横坐标为进化代数 t, 纵坐标为目标函数值, 纵轴起点为实例的已知最优解, 对于每个问题实例, 在 500 进化代数以内, 每方案每代取最优解的平均值绘制进化曲线. 方案 1 使用标准 DE, 方案 4 使用 DDE 与自适应参数调整, 除 j3013_9 以外, 在 j3013_2、3、4、6、8、10 中, 进化后期方案 4 收敛性能优于方案 1, 而在 j3013_1、5 中, 方案 4 收敛比方案 1 差, 综合表 2 中方案 4 平均偏差率优于 1 的评测结果, 表明方案 4 中使用自适应参数控制, 取得了较好的收敛性能; 方案 5 采用了双向调度, 收敛性能显著优于方案 1 和方案 4, 并搜索到了更优解, 表明双向调度比正向调度具有更快的收敛速度; 方案 8 的双向调度同时采用了标准化编码调整及避免局部最优策略, 收敛速度最快, 解的质量最好, 性能也最稳定, 除 j3013_5、6、9 以外, 500 代内均求出了最优解. 在 j3013_9 中, 进化初期方案 8 收敛速度比方案 1 快, 后期略低于方案 1. 由此分析得出, 方案 8 的收敛性能表现最佳.

4.3 算法的结果

基于第 4.1 节与第 4.2 节的分析, 方案 8 可总结为一种使用自适应 DDE 的双向调度算法 (Adaptive dynamic differential evolution based bidirectional scheduling, ADDE-BS), ADDE-BS 基于 DDE, 是综合使用参数自适应控制、标准化编码调整及避免局部最优等策略的双向调度算法, 能获得最小的平均偏差率与较好的算法效率, 其综合性能最佳.

如表 4 所示, 采用 ADDE-BS 算法, 针对 J30、

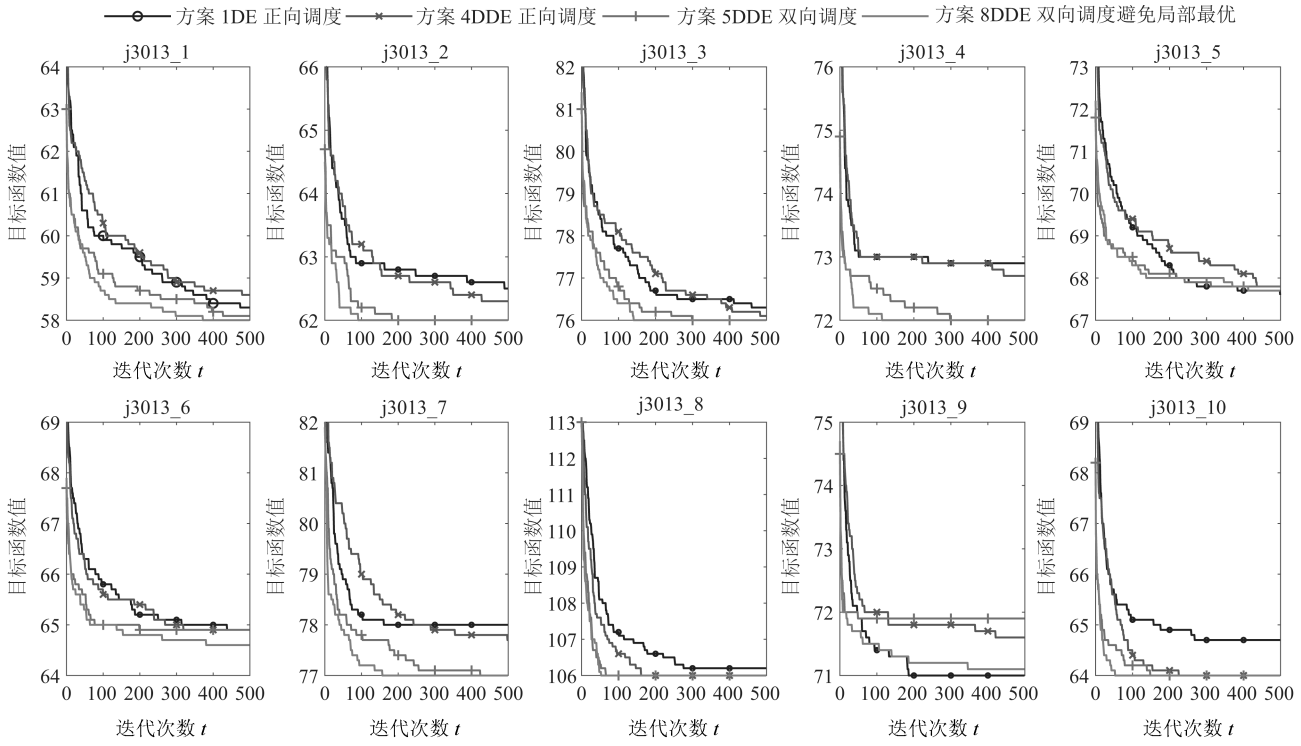


图 4　J3013 不同 DE 方案收敛性能比较

Fig. 4　Comparison of the convergent performance by different DE schemes for J3013

表 4　ADDE-BS 算法结果

Table 4　Results of ADDE-BS algorithm

最大调度次数	5 000				50 000			
实例集	J30	J60	J90	J120	J30	J60	J90	J120
av_dev (%)	0.04	1.32	2.88	25.37	**0.01**	1.03	2.62	24.43
$success$ (%)	97.44	79.58	76.67	30.00	**99.35**	83.75	77.50	31.50
$CPUTime$ (s)	0.40	6.78	16.24	86.92	**0.93**	66.55	179.89	1 004.36

J60、J90、J120 四个系列全部实例, 按最大调度次数不超过 5 000 与 50 000, 分别统计平均偏差率 (保留到小数点后两位)、成功率以及平均 CPU 运行时间. 结果显示, 最大调度次数不超过 50 000 时, J30 平均偏差率很小且不超过 0.01 %, 对于 99 % 以上的问题实例, 算法都能稳定求得 PSPLIB 给出的标准最优解. 比较性能指标可发现, 算法性能随着问题规模的加大而下降, 对于 J30、J60 等中小规模问题, 调度 50 000 次比调度 5 000 次具有更好的解质量, 而对于 J90、J120 等大规模问题, 调度 50 000 次与调度 5 000 次的解质量差异较小, 表明 J90、J120 问题在超过 5 000 次调度以后, 算法获得改进解的能力基本达到极限, 开始产生冗余搜索, 并且消耗大量的 CPU 时间.

4.4　与其他文献算法性能比较

表 5 中收集了最近几年其他文献的 RCPSP 算法平均偏差率, 算法覆盖了遗传、差分进化、粒子群、蚁群、多 Agent 优化等.

对于 J30 实例集, 目前研究中平均偏差率已极低基本接近于零[6]. 表 4 中各种 RCPSP 算法研究结果显示, 5 000 次与 50 000 次调度计算规模以内, 本文 J30 问题平均偏差率 0.01 已处于排行榜前列, 值得注意的是, 同样使用 DE 算法的文献 [18] 中, J30 的平均偏差率为 0.00, 但由于其仅测试了部分实例 (16 个 J30, J60、J90 及 J120 各 15 个), 而且没有明确给出这些实例代号, 因此, DE 算法效果比较仅供参考. 50 000 次调度计算规模以内, 其他文献中, J60 问题解偏差率介于 10 % ∼ 12 % 之间, J120 问题解偏差率介于 26 % ∼ 32 % 之间, 本文的 J60、J90、J120 平均偏差率已显著低于其他文献结果. 由此表明, 对于不同规模的 RCPSP, ADDE-BS 具有较好的泛化求解能力, 并显示出一定的优越性.

表 5 与其他文献算法平均偏差率 (%) 比较

Table 5 Comparison of average deviation rate (%) with art-of-state algorithms

	J30		J60		J90		J120	
	5 000	50 000	5 000	50 000	5 000	50 000	5 000	50 000
本文 ADDE−BS	**0.04**	**0.01**	**1.32**	**1.03**	**2.88**	**2.62**	**25.37**	**24.43**
DE[18] 2016	0.00	0.00	0.98	2.07	4.04	8.81	19.62	31.68
ACO CRO[6] 2017	–	–	11.40	11.40	12.21	12.21	26.53	26.51
COAs[14] 2017	0.00	0.00	10.77	10.58	–	–	32.35	31.23
GA-Part[9] 2017	0.07	0.01	11.08	10.71	–	-	33.36	31.81
Heuristic[24] 2017	0.09	0.03	11.31	10.91	–	–	34.08	32.52
ReVNS[10] 2016	0.01	0.00	11.10	10.88	–	–	33.36	32.21
H-RPSO[7] 2016	0.03	0.01	10.23	10.11	–	–	31.94	30.25
GA-MBX[25] 2013	0.04	0.00	10.94	10.65	–	–	32.89	31.30
MAOA[26] 2015	0.06	0.01	10.84	10.64	–	–	32.64	31.02
PSO-HH[13] 2014	0.04	0.01	11.13	10.68	–	–	32.59	31.23
HGA[27] 2013	0.07	0.01	11.14	10.63	–	–	32.75	30.66
ASH[27] 2013	0.11	0.03	11.33	10.85	–	–	33.54	31.97

5 结论

本文提出一种改进的动态差分进化参数控制及双向调度算法求解 RCPSP. 采用动态差分进化参数自适应控制, 克服了难以确定最佳变异与交叉参数的缺点, 提高了算法的收敛性能; 采用满足时序约束的任务优先数编码, 进化过程中采用交替正反向调度与标准化编码调整策略, 提高了优质解的开发能力, 加速了寻优进程; 采用基于精英保留的种群随机重建策略避免了算法局部收敛, 进一步提高了算法的求解质量与鲁棒性. 使用 PSPLIB 标准问题实例库对文中算法进行了大量测试, 验证了其有效性. 经与其他文献比较, 本文提出的 RCPSP 求解算法具有一定的优越性. 如何结合多种元启发式算法, 提高大规模 RCPSP 的求解性能, 是下一步的研究方向.

References

1 Kolisch R, Hartmann S. Experimental investigation of heuristics for resource-constrained project scheduling: an update. *European Journal of Operational Research*, 2006, **174**(1): 23−37

2 Kolish R, Sprecher A. PSPLIB-A project scheduling problem library. *European Journal of Operational Research*, 1997, **96**(1): 205−216

3 Hartmann S. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 2002, **49**(5): 433−448

4 Deng Lin-Yi, Lin Yan, Jin Chao-Guang. Priority rule-based particle swarm optimization for RCPSP. *Computer Engineering and Applications*, 2009, **45**(10): 40−44
(邓林义, 林焰, 金朝光. 采用优先规则的粒子群算法求解 RCPSP. 计算机工程与应用, 2009, **45**(10): 40−44)

5 Dai Yue-Ming, Tang Ji-Tao, Ji Zhi-Cheng. Cooperative shock search particle swarm optimization with chaos for resource-constrained project scheduling problems. *Journal of Computer Applications*, 2014, **34**(6): 1798−1802
(戴月明, 汤继涛, 纪志成. 协同震荡搜索混沌粒子群求解资源受限项目调度问题. 计算机应用, 2014, **34**(6): 1798−1802)

6 Gonzalez-Pardo A, Del Ser J, Camacho D. Comparative study of pheromone control heuristics in ACO algorithms for solving RCPSP problems. *Applied Soft Computing*, 2017, **60**: 241−255

7 Munlin M, Anantathanavit M. Hybrid radius particle swarm optimization. In: Proceedings of the 2016 IEEE Region 10 Conference (TENCON). Singapore: IEEE, 2016. 2180−2184

8 Valls V, Ballestín F, Quintanilla S. Justification and RCPSP: a technique that pays. *European Journal of Operational Research*, 2005, **165**(2): 375−386

9 Zamani R. An evolutionary implicit enumeration procedure for solving the resource-constrained project scheduling problem. *International Transactions in Operational Research*, 2017, **24**(6): 1525−1547

10 Paraskevopoulos D C, Tarantilis C D, Ioannou G. An adaptive memory programming framework for the resource-constrained project scheduling problem. *International Journal of Production Research*, 2016, **54**(16): 4938−4956

11 Huang Zhi-Yu. Quantum-inspired evolutionary algorithm for resources constrainted project scheduling problem. *Computer Integrated Manufacturing Systems*, 2009, **15**(9): 1779−1787, 1822
(黄志宇. 具有资源约束的项目调度问题中的量子进化算法. 计算机集成制造系统, 2009, **15**(9): 1779−1787, 1822)

12 Lu Zhi-Qiang, Liu Xin-Yi. Algorithm for resource-constrained project scheduling problem with resource transfer time. *Acta Automatica Sinica*, 2018, **44**(6): 1028−1036
(陆志强, 刘欣仪. 考虑资源转移时间的资源受限项目调度问题的算法. 自动化学报, 2018, **44**(6): 1028−1036)

13 Koulinas G, Kotsikas L, Anagnostopoulos K. A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences*, 2014, **277**: 680−693

14 Elsayed S, Sarker R, Ray T, Coello C C. Consolidated optimization algorithm for resource-constrained project scheduling problems. *Information Sciences*, 2017, **418**−**419**: 346−362

15 Wu Liang-Hong, Wang Yao-Nan. *Dynamic Differential Evolution Algorithm and Its Applications*. Beijing: Science Press, 2014. 1−7
(吴亮红, 王耀南. 动态差分进化算法及其应用. 北京: 科学出版社, 2014. 1−7)

16 Das S, Mullick S S, Suganthan P N. Recent advances in differential evolution-an updated survey. *Swarm and Evolutionary Computation*, 2016, **27**: 1−30

17 Cheng M Y, Tran D H, Wu Y W. Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems. *Automation in Construction*, 2014, **37**: 88−97

18 Ali I M, Elsayed S M, Ray T, Sarker R A. A differential evolution algorithm for solving resource constrained project scheduling problems. In: Artificial Life and Computational Intelligence. Lecture Notes in Computer Science, Vol. 9592. Cham: Springer, 2016. 209−220

19 Ciupe A, Meza S, Orza B. Heuristic optimization for the resource constrained Project Scheduling Problem: a systematic mapping. In: Proceedings of the 2016 Federated Conference on Computer Science and Information Systems. Gdansk, Poland: IEEE, 2016. 619−626

20 Brest J, Greiner S, Boskovic B, Mernik M, Zumer V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 2006, **10**(6): 646−657

21 Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 2009, **13**(2): 398−417

22 Zhang J Q, Sanderson A C. JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 2009, **13**(5): 945−958

23 Qing A Y. Dynamic differential evolution strategy and applications in electromagnetic inverse scattering problems. *IEEE Transactions on Geoscience and Remote Sensing*, 2006, **44**(1): 116−125

24 Chand S, Singh H K, Ray T. A heuristic algorithm for solving resource constrained project scheduling problems. In: Proceedings of IEEE Congress on Evolutionary Computation. San Sebastian, Spain: IEEE, 2017. 225−232

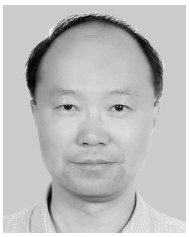
25 Zamani R. A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem. *European Journal of Operational Research*, 2013, **229**(2): 552−559

26 Zheng X L, Wang L. A multi-agent optimization algorithm for resource constrained project scheduling problem. *Expert Systems with Applications*, 2015, **42**(15−16): 6039−6049

27 Lim A, Ma H, Rodrigues B, Tan S T, Xiao F. New meta-heuristics for the resource-constrained project scheduling problem. *Flexible Services and Manufacturing Journal*, 2013, **25**(1−2): 48−73

项 前 东华大学机械工程学院副教授. 主要研究方向为计算机集成制造系统, 计算智能. E-mail: xqsir@dhu.edu.cn
(**XIANG Qian** Associate professor at the College of Mechanical Engineering, Donghua University. His research interest covers computer integrated manufacturing system and computational intelligence.)

周亚云 东华大学机械工程学院硕士研究生. 主要研究方向为计算机集成制造系统. 本文通信作者.
E-mail: zhouyayun@mail.dhu.edu.cn
(**ZHOU YA-Yun** Master student at the College of Mechanical Engineering, Donghua University. Her main research interest is computer integrated manufacturing system. Corresponding author of this paper.)



吕志军 东华大学机械工程学院副教授. 主要研究方向为专家系统, 智能检测.
E-mail: lvzj@dhu.edu.cn
(**LV Zhi-Jun** Associate professor at the College of Mechanical Engineering, Donghua University. His research interest covers expert system and intelligent detection.)